TXE: được clear về 0 (not empty = có dữ liệu đang ở trong TX buffer) khi ta thực hiện phép ghi dữ liệu vào thanh ghi DR. Sau đó dữ liệu từ thanh ghi DR sẽ được đẩy ra TX buffer, rồi từ khoảnh khắc dữ liệu được đẩy từ TX buffer ra thanh ghi dịch (shift register) thì cờ TXE lại được set lên 1 (empty = không có dữ liệu trong TX buffer).

RXNE (receive not empty): được set lên 1 khi có dữ liệu ở trong RX buffer (RXNE được set lên 1 khi truyền xong byte/halfword). Được clear về 0 khi ta thực hiện phép đọc thanh ghi DR.

**Qúa trình truyền/nhận (mastermode):**

Full duplex - continuous:

Trong quá trình ghi byte đầu tiên vào thanh ghi DR (DR sẽ đẩy dữ liệu ra thẳng TX buffer và lúc này TXE = 0) thì TX buffer đẩy 1 byte/halfword song song ra thanh ghi dịch (sau khi đẩy song song ra shift register thì TXE = 1), trong thời gian này (thời gian đẩy dữ liệu song song từ TX ra shift) bit đầu tiên đã được chân MOSI truyền đi. Ngay sau khi đẩy dữ liệu song song từ TX ra shift thì bit TXE cũng được set lên 1, báo hiệu TX buffer đang trống và có thể ghi tiếp vào thanh ghi DR.

Lưu ý: trước khi ghi dữ liệu vào DR thì cần đảm bảo cờ TXE được set lên 1, nếu ko thì dữ liệu sẽ ghi đè lên dữ liệu đang có.

Ngay khi bit đầu tiên trên chân MOSI được truyền, cùng lúc đó tại chân MISO cũng lần lượt đẩy các bit nhận được một cách nối tiếp vào shift register. Sau khi đã truyển đủ 8 bit và nhận đủ 8 bit, lúc này, 8 bit trong shift register là 8 bit nhận được từ chân MISO sẽ được đẩy song song vào RX buffer (thanh ghi DR). Lúc này, ta cần xét cờ RXNE. Nếu cờ đc set = 1 (bởi phần cứng) thì đồng nghĩa đã có dữ liệu nhận được đang nằm trong DR đợi ta thực hiện lệnh đọc. Sau khi đọc, bit RXNE = 0, đòng thời hardware sẽ đẩy dữ liệu từ TX buffer ra shift register, lúc này bit TXE từ 0 lên 1 (vì dữ liệu trong TX vừa được đẩy ra shift register). Khi này, ta lại có thể ghi dữ liệu thứ 3 vào thanh ghi DR và tiếp tục thực hiện các bước y như trên.

Sau khi ghi byte cuối cùng vào thanh ghi DR thì ta cần phải đợi cho đến khi bit TXE được set lên 1 (điều này có nghĩa là dữ liệu trong TX buffer đã được đẩy ra shift register), sau đó ta đợi cờ BSY reset về 0 (điều này có nghĩa toàn bộ dữ liệu cuối cùng đã được truyền đi, và SPI không còn truyền hay nhận gì hết). Sau khi 2 điều kiện trên được thỏa mãn, ta đã có thể out function và làm việc khác.

**Qúa trình truyền/nhận (slavemode):**

Quy trình tương tự như master mode, chỉ khác 1 chỗ là cờ BSY bị kéo xuống mức thấp mỗi lần truyền/ nhận xong 1 data.

**Thủ tục kết thúc truyền và nhận:**

* Trong hàm Trans&Receive(). Sau khi đẩy đủ n byte và DR thì ta đã nhận được 1 byte, chính vì vậy ta vẫn cứ đợi cho cờ RXNE == 1 rồi read, khi count(length) == 0 thì tức là ta đã đọc đủ 3 byte cũng như truyền đủ 3 byte. Việc cần làm bây giờ là đợi cờ BSY == 0.

+ Hàm này chỉ áp dụng cho full-duplex.

* Trong hàm Transmit(). Sau khi đẩy đủ n byte vào DR thì ta cần đợi cho cờ TXE == 1. Điều này có nghĩa, byte cuối cùng đã được đẩy ra shift register. Sau đó ta cần đợi cho cờ BSY == 0. Điều này chỉ ra quá trình truyền/nhận đã kết thúc.
* Trong hàm Receive().

+ SPI ở mode master và đang cấu hình direction 2 line receive only hoặc 1 line bidirectional receive thì sau khi đợi RXNE cuối cùng == 1 thì ta đọc thanh ghi DR, sau đó đợi 1 vòng lặp xung clock (có thể 1 lệnh if hoặc vòng lặp for) rồi sau đó disable SPI (clear SPE bit).

+ Nếu là ở mode master và receive only. Điều ta cần quan tâm là dữ liệu nhân được, chính vì vậy trước khi thoát khỏi hàm thì ta phải check thêm liệu bit RXNE đã == 0 hay chưa.

Đọc trang 718 để biết thêm thông tin disable SPI.

**A screen shot of a computer

Description automatically generatedA table with text and symbols

Description automatically generated with medium confidenceCấu hình GPIO:**

Giải thích tại sao khi debug thanh ghi DR thì data được khi xuống thanh ghi không phải là giá trị vừa mới đc ghi:

+ Dữ liệu sau khi ghi vào thanh ghi DR, ngay lập tức được gửi song song ra TX buffer, rồi sau đó TX buffer cũng ngay lập tức dịch từng bit ra chân MOSI, đồng thời thanh ghi DR lúc này sẽ đc dịch giá trị từ RX buffer. Chính vì vậy giá trị trong DR lúc này chính là những gì nằm trong thanh ghi DR của slave.

**Lý do tại sao trong struct Handling\_SPI, thư viện HAL lại tạo một vài con trỏ chứa gia trị Tx, Rx, Length:**

* Khi sử dụng ngắt, việc thoát khỏi vector ISR của ngắt sẽ khiến mất đi dữ liệu biến nếu biến đó được tạo ra từ bên trong hàm ngắt. Chính vì vậy, người ta tạo ra một biến trong struct, nhằm lưu lại giá trị ngay cả khi thoát khỏi hàm ngắt.

**Procedures for SPI\_Receive\_Interrupt:**

1. Gọi hàm cấu hình IQR để set bit tương ứng trong thanh ghi NVIC, cũng như enable bit RXNEIE.
2. Gọi hàm cấu hình mức ưu tiên ngắt (interrupt priority).
3. Starting:
   1. Với mục tiêu STM32 là master sẽ nhảy vào ngắt khi có dữ liệu từ slave gửi đến(master receive), ta cần **Transmit xuống slave** dummy bytes. Đồng thời ta phải **tạo 1 biến global** để lưu dữ liệu received, vì nếu tạo biến trong hàm ngắt thì biết đó sẽ bị xóa sau khi hàm ngắt bị kết thúc. Ngoài ra, trong hàm ngắt receive ta cũng cần thực hiện lệnh **read từ thanh ghi DR**, nếu ko chương trình sẽ bị kẹt mãi trong ISR SPI\_IQRHandler.

**Lý do tại sao phải xóa cờ OVRF(RA6M3):**

* Khi cờ OVRF = 1, shift register sẽ không dịch dữ liệu vào Reveive buffer nữa.
* Dữ liệu trước khi xảy ra lỗi (trước khi OVRF = 1) sẽ được tồn tại mãi mãi trong RX Buffer cho đến khi ta Read bit OVRF và sau đó write “0” xuống OVRF.